Project-1

Bias correction of numerical prediction model temperature forecast

Data Set

krishna sai surendra babu kalluri(50569326)

1.Introduction:

This data is for the purpose of bias correction of next-day maximum and minimum air temperatures forecast of the LDAPS model operated by the Korea Meteorological Administration over Seoul, South Korea. This data consists of summer data from 2013 to 2017. The input data is largely composed of the LDAPS model's next-day forecast data, in-situ maximum and minimum temperatures of present-day, and geographic auxiliary variables. There are two outputs (i.e. next-day maximum and minimum air temperatures) in this data. Hindcast validation was conducted for the period from 2015 to 2017.

Reliable forecasting of air temperature at 2 m above the land surface plays a significant role when preparing for potential weather-related disasters, such as heat waves (i.e., maximum daytime air temperature) and coldspells (i.e., minimum night time air temperature). Extreme air temperatures can also cause various social andeconomic problems such as heat-related disease and high energy consumption (Klinenberg, 2015; Russoet al., 2019). In particular, the increasing intensity, frequency and duration of extreme air temperatures during the summer season (Perkins et al., 2012), and the fact that more than half of the Earth's population now lives in cities (Schulze & Langenberg, 2014) suggest that accurate air temperature forecasting is essential for urban areas.

Nature of Dataset:

Data Set Characteristics: Multivariate

Attribute Characteristics: Real

Area: Physical

Number of Instances: 7750

Number of Attributes: 25

Associated Tasks: Regression

Attribute Information:

Attribute	Attribute Information
Station	used weather station number: 1 to 25
Date	Present day yyyy-mm-dd ('2013-06-30' to '2017-08-30')
Present_Tmax	Maximum air temperature between 0 and 21 h on the present day (Â°C): 20 to 37.6
Present_Tmin	Minimum air temperature between 0 and 21 h on the present day (Â°C): 11.3 to 29.9
LDAPS_RHmin	LDAPS model forecast of next-day minimum relative humidity (%): 19.8 to 98.5
LDAPS_RHmax	LDAPS model forecast of next-day maximum relative humidity (%): 58.9 to 100
LDAPS_Tmax_lapse	LDAPS model forecast of next-day maximum air temperature applied lapse rate (Â°C): 17.6 to 38.5
LDAPS_Tmin_lapse	LDAPS model forecast of next-day minimum air temperature applied lapse rate (Â°C): 14.3 to 29.6
LDAPS_WS	LDAPS model forecast of next-day average wind speed (m/s): 2.9 to 21.9
LDAPS_LH	LDAPS model forecast of next-day average latent heat flux (W/m2): -13.6 to 213.4
LDAPS_CC1	LDAPS model forecast of next-day 1st 6-hour split average cloud cover (0-5 h) (%): 0 to 0.97
LDAPS_CC2	LDAPS model forecast of next-day 2nd 6-hour split

	average cloud cover (6-11 h) (%): 0 to 0.97
LDAPS_CC3	LDAPS model forecast of next-day 3rd 6-hour split average cloud cover (12-17 h) (%): 0 to 0.98
LDAPS_CC4	LDAPS model forecast of next-day 4th 6-hour split average cloud cover (18-23 h) (%): 0 to 0.97
LDAPS_PPT1	LDAPS model forecast of next-day 1st 6-hour split average precipitation (0-5 h) (%): 0 to 23.7
LDAPS_PPT2	LDAPS model forecast of next-day 2nd 6-hour split average precipitation (6-11 h) (%): 0 to 21.6
LDAPS_PPT3	LDAPS model forecast of next-day 3rd 6-hour split average precipitation (12-17 h) (%): 0 to 15.8
LDAPS_PPT4	LDAPS model forecast of next-day 4th 6-hour split average precipitation (18-23 h) (%): 0 to 16.7
lat	Latitude (Â°): 37.456 to 37.645
lon	Longitude (Â°): 126.826 to 127.135
DEM	Elevation (m): 12.4 to 212.3
Slope	Slope (Â°): 0.1 to 5.2
Solar radiation	Daily incoming solar radiation (wh/m2): 4329.5 to 5992.9
Next_Tmax	The next-day maximum air temperature (Â°C): 17.4 to 38.9
Next_Tmin	The next-day minimum air temperature (Â°C): 11.3 to 29.8

2.Methods :

This is a regression model which is to made bias correction of next-day maximum and minimum air temperatures.

This Dataset does not require pre-processing.

steps to be implement:

1)First, we need to import the data set

2)split data: Now we split the data into training set and test set

3)Then we will do model implementation .The methods we used for implementation are

1)simple linear regression

2)multiple linear regression

3)KNN-regression

simple linear regression:

It is a statistical method that allows us to summarize and study relationships between two continuous (quantitative) variables. One variable denoted x is regarded as an independent variable and other one denoted y is regarded as a dependent variable. It is assumed that the two variables are linearly related. Hence, we try to find a linear function that predicts the response value(y) as accurately as possible as a function of the feature or independent variable(x).

In this method, we predict the next_Tmax using present_Tmax and next_Tmin using present_Tmin .This model can be observed using MSE Error .

```
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
#Tmax Prediction

slr_trainingModel <- lm(Next_Tmax ~ Present_Tmax, data = trainingData)
slr_NextTmaxPredict <- predict(slr_trainingModel, testData)  # predict distance
slr_NextTmaxPredict

summary(slr_trainingModel)

AIC (slr_trainingModel)
```
```

```
slr_actuals_preds <- data.frame(cbind(actuals=testData$Next_Tmax,
predicteds=slr_NextTmaxPredict))  # make actuals_predicteds dataframe.

slr_correlation_accuracy <- cor(slr_actuals_preds)  # 82.7%

head(slr_actuals_preds)

slr_MSE <- mean((testData$Next_Tmax-slr_NextTmaxPredict)^2, na.rm = TRUE)

slr_MSE

#Tmin Prediction


slr_trainingModelTmin <- lm(Next_Tmin ~ Present_Tmin , data = trainingData)
slr_NextTminPredict <- predict(slr_trainingModelTmin, testData)  # predict distance
slr_NextTminPredict

summary(slr_trainingModelTmin)

AIC (slr_trainingModelTmin)

slr_actuals_preds_Tmin <- data.frame(cbind(actuals=testData$Next_Tmin,
predicteds=slr_NextTminPredict))  # make actuals_predicteds dataframe.

correlation_accuracy <- cor(slr_actuals_preds_Tmin)  # 82.7%

head(slr_actuals_preds_Tmin)

slr_MSETmin <- mean((testData$Next_Tmin-slr_NextTminPredict)^2, na.rm = TRUE)

slr_MSETmin
```

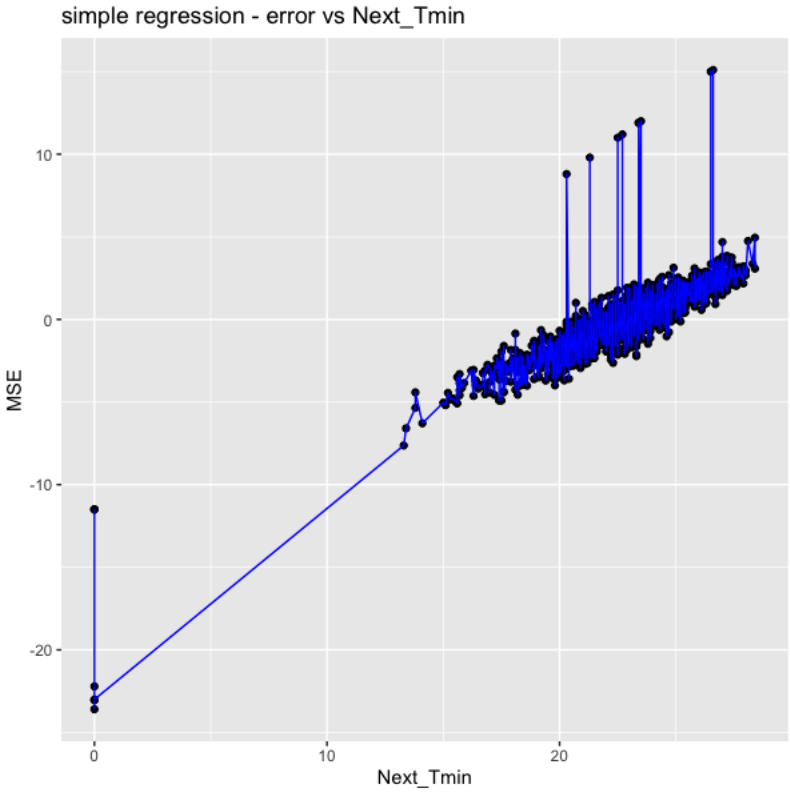
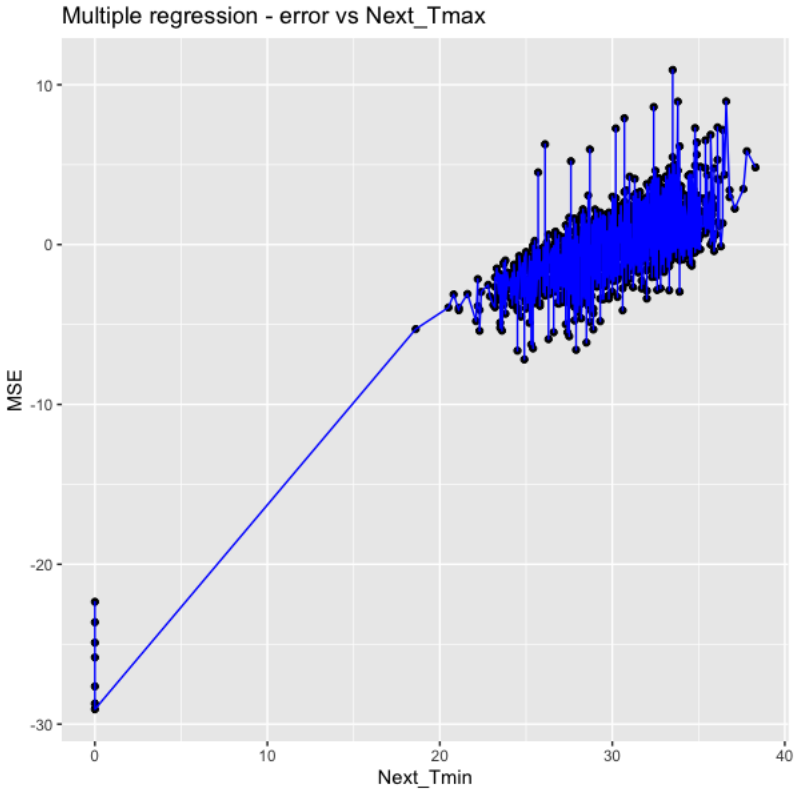
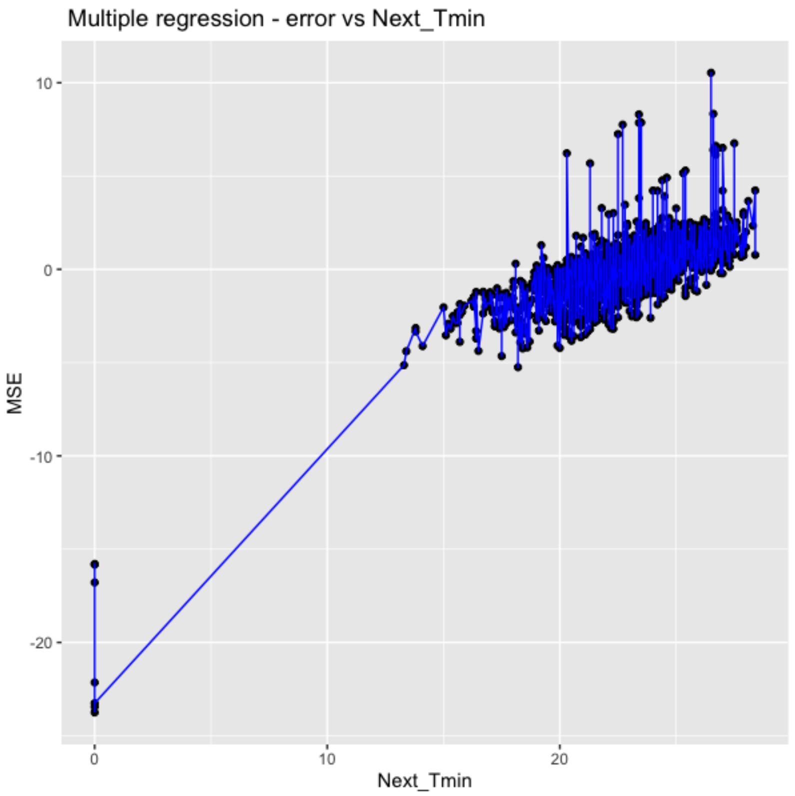
Multiple linear regression:

It is the most common form of Linear Regression. Multiple Linear Regression basically describes how a single response variable Y depends linearly on a number of predictor variables.

In this model we use the scatter plot to decide what are the variables that have effect on predictor and predict the next_Tmax , next_Tmin using present_Tmax , present_Tmin but we predict with multiple variables.

```{r setup, include=FALSE}
```

````
knitr::opts_chunk$set(echo = TRUE)
trainingModel <- lm(Next_Tmax ~ Present_Tmax + LDAPS_RHmax + LDAPS_Tmax_lapse +
LDAPS_Tmin_lapse + lat + lon + Slope , data = trainingData)
NextTmaxPredict <- predict(trainingModel, testData)  # predict distance
NextTmaxPredict

summary(trainingModel)

AIC (trainingModel)

actuals_preds <- data.frame(cbind(actuals=testData$Next_Tmax,
predicteds=NextTmaxPredict))  # make actuals_predicteds dataframe.

correlation_accuracy <- cor(actuals_preds)  # 82.7%

head(actuals_preds)

MSETmax <- mean((testData$Next_Tmax-NextTmaxPredict)^2, na.rm = TRUE)

MSETmax

#Tmin Prediction

trainingModelTmin <- lm(Next_Tmin ~ Present_Tmin + LDAPS_RHmin + LDAPS_Tmax_lapse
+ LDAPS_Tmin_lapse + lat + lon + Slope , data = trainingData)
NextTminPredict <- predict(trainingModelTmin, testData)  # predict distance
NextTminPredict

summary(trainingModelTmin)

AIC (trainingModelTmin)

actuals_preds_Tmin <- data.frame(cbind(actuals=testData$Next_Tmin,
predicteds=NextTminPredict))  # make actuals_predicteds dataframe.

correlation_accuracy <- cor(actuals_preds_Tmin)  # 82.7%

head(actuals_preds_Tmin)

MSETmin <- mean((testData$Next_Tmin-NextTminPredict)^2, na.rm = TRUE)

MSETmin
```
````

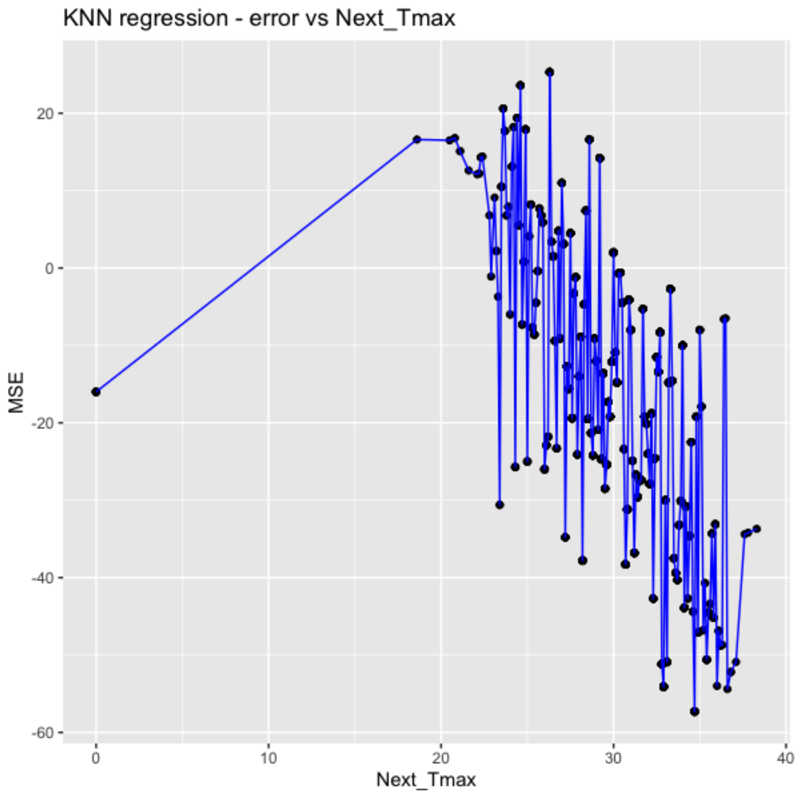
KNN regression:
````

KNN which stand for K Nearest Neighbor is a Supervised Machine Learning algorithm that classifies a new data point into the target class, depending on the features of its neighboring data points.

In this method we will predict the next_Tmax with Present_Tmax based on nearest k-elements for the testing present_Tmax.

````
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
#KNN

x_trn_bias = trainingData$Present_Tmax
y_trn_bias = trainingData$Next_Tmax
x_tst_tmax_bias = testData$Present_Tmax
y_tst_tmax_bias = testData$Next_Tmax
x_tst_tmin_bias = testData$Present_Tmin
y_tst_tmin_bias = testData$Next_Tmin

x_trn_bias_min = min(x_trn_bias, na.rm = TRUE)
x_trn_bias_max = max(x_trn_bias, na.rm = TRUE)

lstat_grid = data.frame(lstat = seq(x_trn_bias, x_trn_bias,
                            by = 0.01))

pred_tmax_005 = knn(data.frame(x_trn_bias), data.frame(y_tst_tmax_bias), y_trn_bias,
k=5)

pred_tmin_005 = knn(data.frame(x_trn_bias), data.frame(y_tst_tmin_bias), y_trn_bias, k=5)

pred_tmax_005 <- as.numeric(pred_tmax_005)

pred_tmax_005

pred_tmin_005 <- as.numeric(pred_tmin_005)

pred_tmin_005

MSE_KNN_Tmax = mean((testData$Next_Tmax-as.numeric(pred_tmax_005))^2, na.rm =
TRUE)

MSE_KNN_Tmin = mean((testData$Next_Tmin-as.numeric(pred_tmin_005))^2, na.rm =
TRUE)

MSE_KNN_Tmax_Individual = (testData$Next_Tmax - as.numeric(pred_tmax_005))
```
````

3.Results:

simple linear regression:

Here after creating model we fill get the results as follows:

```
> summary(slr_trainingModel)

Call:
lm(formula = Next_Tmax ~ Present_Tmax, data = trainingData)

Residuals:
    Min      1Q  Median      3Q     Max 
-32.082  -1.715   0.347   1.900  16.832 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  18.367710   0.290219   63.29   <2e-16 ***
Present_Tmax  0.400996   0.009743   41.16   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.163 on 6199 degrees of freedom
Multiple R-squared:  0.2146,	Adjusted R-squared:  0.2145 
F-statistic:  1694 on 1 and 6199 DF,  p-value: < 2.2e-16
 slr_correlation_accuracy <- cor(slr_actuals_preds)  # 82.7%

 head(slr_actuals_preds)
   actuals predicteds
8     31.1   31.23967
11    31.2   31.07927
12    32.6   31.03917
24    31.3   30.99907
34    27.1   30.59808
39    27.9   30.71838

 slr_MSE
[1] 10.29969

summary(slr_trainingModelTmin)

Call:
lm(formula = Next_Tmin ~ Present_Tmin, data = trainingData)
```

```
Residuals:
     Min       1Q   Median       3Q      Max 
-24.7330  -1.1517   0.1222   1.1546  16.0002 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  11.49979    0.20514   56.06   <2e-16 ***
Present_Tmin  0.49378    0.00882   55.98   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.277 on 6199 degrees of freedom
Multiple R-squared:  0.3358,  Adjusted R-squared:  0.3357 
F-statistic:  3134 on 1 and 6199 DF,  p-value: < 2.2e-16

 head(slr_actuals_preds_Tmin)
   actuals predicteds
8     22.9   23.15290
11    24.5   23.10352
12    22.2   22.46161
24    23.7   23.84418
34    21.0   22.21472
39    21.5   22.80726

slr_MSETmin
[1] 5.395084
```

we will get the accuracy rate as 82.7 and MSE error is 10.29 ,5,29 .After this we will plot the graph between MSE error and next_Tmax,Next_Tmin

````
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)

#Tmax

plot_Tmax <- data.frame(testData$Next_Tmax,slr_MSE_Tmax_Individual)
ggplot(plot_Tmin,aes(x= testData$Next_Tmax,y= slr_MSE_Tmax_Individual))+
  geom_point() +
  geom_line(color="blue") +
  xlab("Next_Tmax") +
  ylab("MSE") +
  ggtitle("simple regression - error vs Next_Tmax")

#Tmin

plot_Tmin <- data.frame(testData$Next_Tmin,slr_MSE_Tmin_Individual)
````
````

```
ggplot(plot_Tmin,aes(x= testData$Next_Tmin,y= slr_MSE_Tmin_Individual))+
  geom_point() +
  geom_line(color="blue") +
  xlab("Next_Tmin") +
  ylab("MSE") +
  ggtitle("simple regression - error vs Next_Tmin")
```



2)Multiple Linear regression:

After creating model we will get the results as follows:

```
Call:
lm(formula = Next_Tmax ~ Present_Tmax + LDAPS_RHmax + LDAPS_Tmax_lapse +
    LDAPS_Tmin_lapse + lat + lon + Slope, data = trainingData)

Residuals:
    Min      1Q  Median      3Q     Max 
-32.807  -1.241   0.129   1.364  10.356 

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)    
(Intercept)      204.580075  55.021374   3.718 0.000202 ***
Present_Tmax       0.224973   0.009212  24.422  < 2e-16 ***
LDAPS_RHmax       -0.124361   0.003894 -31.940  < 2e-16 ***
LDAPS_Tmax_lapse   0.516322   0.014579  35.415  < 2e-16 ***
LDAPS_Tmin_lapse  -0.082367   0.020515  -4.015 6.01e-05 ***
lat                3.019484   0.708810   4.260 2.08e-05 ***
lon               -2.337874   0.441628  -5.294 1.24e-07 ***
Slope              0.121873   0.025425   4.793 1.68e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.634 on 6193 degrees of freedom
Multiple R-squared:  0.4558,  Adjusted R-squared:  0.4552 
F-statistic:   741 on 7 and 6193 DF,  p-value: < 2.2e-16
```

```
 head(actuals_preds)
   actuals predicteds
8     31.1   31.66598
11    31.2   30.72283
12    32.6   31.20583
24    31.3   32.23777
34    27.1   28.29261
39    27.9   29.08143

MSETmax
[1] 7.61941

Call:
lm(formula = Next_Tmin ~ Present_Tmin + LDAPS_RHmin + LDAPS_Tmax_lapse +
    LDAPS_Tmin_lapse + lat + lon + Slope, data = trainingData)

Residuals:
     Min       1Q   Median       3Q      Max
-25.7176  -0.8555   0.0597   0.9256   9.9824

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)      -21.518997  41.232408  -0.522  0.60176
Present_Tmin       0.325483   0.008530  38.157  < 2e-16 ***
LDAPS_RHmin       -0.064847   0.002588 -25.058  < 2e-16 ***
LDAPS_Tmax_lapse  -0.413929   0.016468 -25.135  < 2e-16 ***
LDAPS_Tmin_lapse   0.844580   0.022731  37.156  < 2e-16 ***
lat                1.719444   0.532440   3.229  0.00125 **
lon               -0.248604   0.330937  -0.751  0.45255
Slope              0.025870   0.018915   1.368  0.17146
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.974 on 6193 degrees of freedom
Multiple R-squared:  0.5011,  Adjusted R-squared:  0.5005
F-statistic: 888.6 on 7 and 6193 DF,  p-value: < 2.2e-16

head(actuals_preds_Tmin)
   actuals predicteds
8     22.9   23.86079
11    24.5   24.12174
12    22.2   23.99420
24    23.7   25.06402
34    21.0   21.61785
39    21.5   22.68796
```

```
 MSETmin <- mean((testData$Next_Tmin-NextTminPredict)^2, na.rm = TRUE)

 MSETmin
[1] 4.3595
```

Here the MSE error is 7.61 and the accuracy rate is 92.4% .After this we will plot MSE error vs Tmin,Tmax

````
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)

#Tmax

plot_Tmax <- data.frame(testData$Next_Tmax,MSE_Tmax_Individual)
ggplot(plot_Tmin,aes(x= testData$Next_Tmax,y= MSE_Tmax_Individual))+
  geom_point() +
  geom_line(color="blue") +
  xlab("Next_Tmin") +
  ylab("MSE") +
  ggtitle("Multiple regression - error vs Next_Tmax")

#Tmin

MSE_Tmin_Individual = (testData$Next_Tmin - as.numeric(NextTminPredict))

plot_Tmin <- data.frame(testData$Next_Tmin,MSE_Tmin_Individual)
ggplot(plot_Tmin,aes(x= testData$Next_Tmin,y= MSE_Tmin_Individual))+
  geom_point() +
  geom_line(color="blue") +
  xlab("Next_Tmin") +
  ylab("MSE") +
  ggtitle(" Multiple regression - error vs Next_Tmin")


```
````





3)KNN regression:

```
MSE_KNN_Tmax = mean((testData$Next_Tmax-as.numeric(pred_tmax_005))^2, na.rm =
TRUE)

MSE_KNN_Tmax
```

```
[1] 610.85

MSE_KNN_Tmin = mean((testData$Next_Tmin-as.numeric(pred_tmin_005))^2, na.rm =
TRUE)
 MSE_KNN_Tmax
 [1] 195.54
```

Here clearly the MSE error is too high and is not a good method to perform .After this we plot garph for MSE vs Tmax,Tmin

````
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
#Tmax

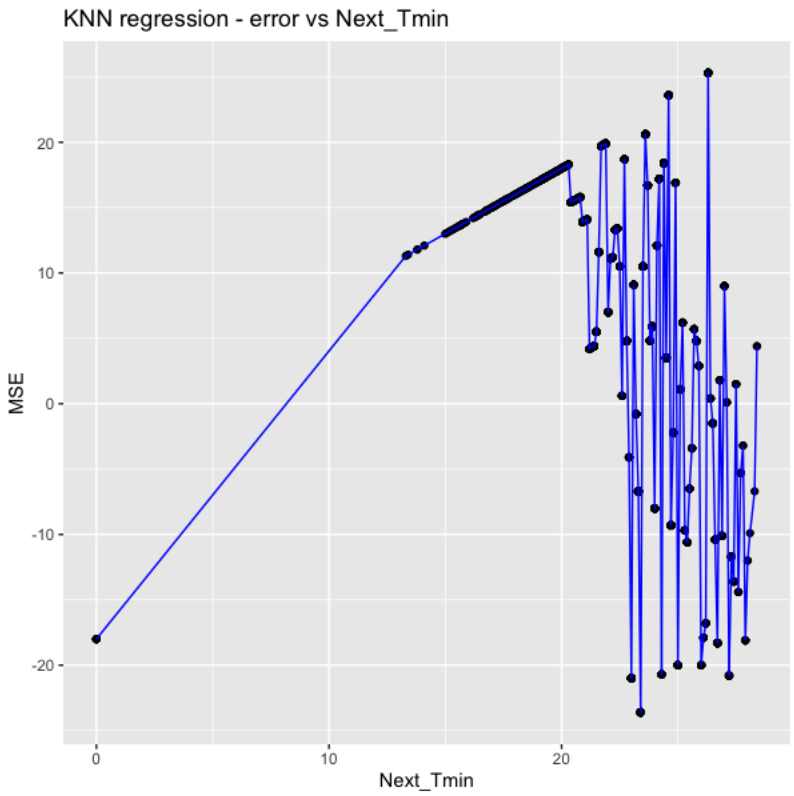
MSE_KNN_Tmax_Individual = (testData$Next_Tmax - as.numeric(pred_tmax_005))
plot_Tmax <- data.frame(y_tst_tmax_bias,MSE_KNN_Tmax_Individual)
ggplot(plot_Tmax,aes(x = y_tst_tmax_bias,y = MSE_KNN_Tmax_Individual)) +
  geom_point() +
  geom_line(color="blue") +
  xlab("Next_Tmax") +
  ylab("MSE") +
  ggtitle("KNN regression - error vs Next_Tmax")


#Tmin

MSE_KNN_Tmin_Individual = (testData$Next_Tmin - as.numeric(pred_tmin_005))

plot_Tmin <- data.frame(y_tst_tmin_bias,MSE_KNN_Tmin_Individual)
ggplot(plot_Tmin,aes(x= y_tst_tmin_bias,y= MSE_KNN_Tmin_Individual))+
  geom_point() +
  geom_line(color="blue") +
  xlab("Next_Tmin") +
  ylab("MSE") +
  ggtitle("KNN regression - error vs Next_Tmin")
```
````





Method Analysis:

After splitting the data into training and test data ,we performed three method to analyse the data and find out the MSE error for the three methods, plotted the graphs .We know that a method with minimum MSE error is the best method for implementation. Among the three Multiple Linear Regression is the best regression with error rate is low with 7.6% and KNN regression is the highest error rate .

4.Discussion:

Since the Multiple linear regression gives the lowest error rate of 7.6% with 93.4% best accuracy rate among all the models .so, we can conclude that Multiple linear Regression is the best classifier for this data set with highly recommended.

In future , also Multiple linear regression remains best for this data set with accuracy rate 93.4%.

5.Reference cited:

Cho, D., Yoo, C., Im, J., & Cha, D. (2020). Comparative assessment of various machine learning-based bias correction methods for numerical weather prediction model forecasts of extreme air temperatures in urban areas. Earth and Space Science.